IN THE TITLE OF THE INVENTION:

Please change the title of invention as follows.

ACCESSING LIBRARY OF ACCESSORY DEVICE

IN THE DISCLOSURE:

Please change the disclosure as follows.

In the paragraph beginning at p. 1, line 13:

--The number of the accessories for mobile devices is increasing. In this description the term accessory means an additional device which brings one or more new features to the mobile device when the accessory is brought into communication connection with the mobile device. Many such accessories need a library or driver through which it is used by applications, such as ~~Java~~JAVA™ applications, running in the mobile device. The library is built to enable access and control of the accessory. Such libraries are often stored in the mobile device for different accessories by *e.g.* the manufacturer of the mobile device. There may exist a number of such libraries stored in the mobile device to assure that different kind of accessories can be used by the mobile device. However, the user may use only limited number of accessories, if any, and different users may use different accessories. Therefore, the libraries which are permanently stored in the mobile device and which are not needed by the user unnecessarily reserve the often quite restricted storage capacity of the mobile device. There exist several techniques to solve this problem, but all have some limitations.--

In the paragraph beginning at p. 2, line 31:

-- If ~~Java~~JAVA™ libraries for all possible (future) accessories is put to the device ~~Java~~JAVA™ platform, this has a negative effect on the memory consumption of the device. In addition, applications need to have ways to identify which accessories are actually in use and which only have a placeholder library available.--

In the paragraph beginning at p. 3, line 8:

--~~Java~~JAVA™ application life cycle consists of user locating the application, downloading the application to a device, installing it, using it, and removing it from the device. Application download typically takes place e.g. over the air from a server, or locally from a PC via cable or wireless link. It is possible to use an accessory for enhancing the mentioned application life cycle phases and improving user experience.--

In the paragraph beginning at p. 3, line 15:

-- The European patent application EP 1 347 623 discloses downloading of application software for an accessory device to a mobile device. The application software is stored on the memory of the accessory. the application software can be platform independent ~~Java~~JAVA™ applets or Symbian applications. The mobile device might comprise a ~~Java~~JAVA™ VM (virtual machine), *e.g.* a KJAVA™™~~kJava~~JAVA™ VM or a MIDP ~~Java~~JAVA™ VM, or a Symbian OS (Operating System). When the accessory is connected to the mobile device it is detected and the downloading of the application software from the accessory is initiated. When the application software is downloaded to the mobile device it can be started and used for controlling the accessory and exchanging information with the accessory. The connection and the information transfer between the mobile device and the accessory is conducted through a smart accessory manager. It has built in application programming interfaces (APIs) for different kinds of connections. By using this downloading method the downloaded application can use the properties of the accessory from which the application was downloaded. This method, however, does not automatically enable other applications running on the mobile device to use the properties of the accessory.--

In the paragraph beginning at p. 6, line 1:

--The invention provides a more convenient mechanism for bringing accessory functionality available for ~~Java~~JAVA™ applications running on the device. Support for accessories does not have to be built in the device in a manufacturing phase.--

In the paragraph beginning at p. 6, line 15:
-- This invention is more lightweight than Jini and also implementable on embedded devices with small amount of memory and is not limited to ~~Java~~JAVA™ technology.--

In the paragraph beginning at p. 7, line 15:
-- In the following the invention will be described using ~~Java~~JAVA™ library as an example of libraries but the invention is not limited to ~~Java~~JAVA™ environment and ~~Java~~JAVA™ libraries only. Another possible environments are, for example, BREW™ from Qualcomm® and .NET™ Framework from Microsoft. It is also assumed that the device 1 has a ~~Java~~JAVA™ platform (typically J2ME™ MIDP; The JAVA™~~Java™~~ 2 platform, Micro Edition, Mobile Information Device Profile) with the additional capability to dynamically install libraries to the device 1. The device 1 can be any device capable of running applications and provided by a connecting means for connecting an accessory 2 with the device 1. Fig. 3 shows as a block diagram of an example of such a device 1. The device 1 comprises a control unit 1.1 for, *inter alia*, controlling the operations of the device 1 and for running applications on the device 1. The control unit 1.1 can be implemented as separate circuits, such as one or more processors, or as an integrated circuit, such as an ASIC (Application Specific Integrated Circuit). The device also comprises a memory 1.2 for storing applications, APIs, libraries, control software, data etc. The memory 1.2 may consist of read-only memory and random access memory. There is also provided an interface 1.3 in the device 1 for providing a connection with an accessory 2. The interface 1.3 may comprise wired connecting means and/or wireless connecting means. In addition to the interface 1.3 the device 1 may comprise communication means 1.4, such as mobile communication means, for

performing communication tasks with a communication network and/or with another device (not shown). The device 1 further comprises a user interface comprising, for example, a display 1.5, a keyboard 1.6, a microphone 1.7 and/or a loudspeaker/headphone 1.8.--

In the paragraph beginning at p. 9, line 21:

--The relationship between the device 1 and the accessory 2 is shown in Fig. 1. In the device 1 the hardware (HW) and the software (SW) can be regarded as providing a platform 4 for the operation of the device 1. The platform 4 of the device 1 comprises a hardware section 4.1, a software section 4.2 including control software and applications of the device 1, standard ~~Java~~JAVA™ libraries 4.3 and generic accessory support library 4.4. The platform 4 of the device 1 also comprises ~~Java~~JAVA™ application section 4.6 for storing ~~Java~~JAVA™ applications, and an accessory library 4.5 for storing libraries 3 downloaded from the accessory 2. There is also provided a standardized ~~Java~~JAVA™ API 4.7, a ~~Java~~JAVA™ accessory library API 4.12 that is implemented by the downloaded accessory library and an internal ~~Java~~JAVA™ API 4.8. The standardized ~~Java~~JAVA™ API 4.7 is used as an interface between standard ~~Java~~JAVA™ libraries and ~~Java~~JAVA™ applications. The ~~Java~~JAVA™ accessory library API 4.12 provides accessory functionality for the application. The internal ~~Java~~JAVA™ API is used as an interface between the accessory library 4.5 and the generic accessory support library 4.4 to access the accessory functionality. The platform 4 of the device 1 also includes an application management software (AMS) 4.9 as a part of the ~~Java~~JAVA™ system that controls execution, installation and removal of ~~Java~~JAVA™ applications. The platform 4 of the device 1 further includes an accessory server 4.10 for controlling at least the detection of attachment and detachment of the accessory 2. The accessory server 4.10 has access to the desired parts of the hardware 4.1 for performing the detection, for example, by examining the status of the interface 1.3.--

In the paragraph beginning at p. 10, line 13:
-- In the installation phase 6 of the library 3 the accessory 2 can, for example, offer the following information to the device 1: library location, name, size, version and vendor. This information is used by the device ~~Java~~JAVA™ platform 4 for installing the library 3.--

In the paragraph beginning at p. 10, line 29:
-- There may be some restrictions so that some (unauthorized) applications are not allowed to use the accessory library 3. This can be managed by, for example, a ~~Java~~JAVA™ platform security settings. If the accessory is to be used *e.g.* with several different phone models with different ~~Java~~JAVA™ platforms, some ~~Java~~JAVA™ platforms may already include the library 3 that the accessory 2 brings with it. ~~Java~~JAVA™ platform should also be able to handle such situations. In one embodiment of the invention the dynamically installed ~~Java~~JAVA™ library 3 is using existing internal ~~Java~~JAVA™ API 4.8. The existing ~~Java~~JAVA™ APIs 4.7, 4.8 also include already installed dynamic libraries.--

In the paragraph beginning at p. 11, line 19:
-- In an other embodiment of the present invention libraries 3 of the accessory are not downloaded to the device 1 but they are made available to the device 1 by using other means. This can be performed, for example, so that the accessory 2 is powered up and connected either by wired or wireless manner with the device 1. When the device 1 has detected the existence of the accessory 2 it performs the similar identification step and discovers whether the accessory 2 comprises a library 3 which the applications of the device 1 can use. The device 1 adds information on such library 3 or libraries to the platform 4. When the library 3 is accessed by an application running on the device 1 the device 1 and the accessory 2 communicate to make the library 3 of the accessory 2 accessible to the application as if the library 3 were installed on the platform 4 of the device 1. This approach may require that the

~~Java~~JAVA™ platform is modified in installation phase to find the library 3 from the accessory 2 during run-time.--

In the paragraph beginning at p. 12, line 3:
--There are several possibilities to implement this invention. One typical scenario would be a mobile phone as a device 1 with an accessory 2 that is attached to the phone. The device 1 has a ~~Java~~JAVA™ platform (typically J2ME MIDP) with the additional capability to dynamically install libraries 3 to the platform 4.--

In the paragraph beginning at p. 12, line 9:
-- In an example implementation the attaching and detaching of accessories 2 dynamically is allowed without requiring restarting of the device 1, the ~~Java~~JAVA™ platform 4 or even the application. In this kind of implementation the library/libraries 3 offered by the accessory 2 are transparently installed to the ~~Java~~JAVA™ platform 4 when the accessory 2 is attached and, respectively, uninstalled from the ~~Java~~JAVA™ platform when the accessory 2 is detached. In one typical implementation the availability of the possible dynamic libraries 3 is checked when the application starts. If the required libraries 3 are available, the application may start executing. One possibility to provide the application information about the availability of the library 3 would be the ~~Java~~JAVA™ platform which is modified so that it comprises means for the applications to ask about the currently available libraries.--
In the paragraph beginning at p. 13, line 33:
-- If the cover 8 has an attribute (e.g. JavaRequiredAPI) informing that the operation of the cover 8 requires a ~~Java~~JAVA™ API application, management software 4.9 checks the attribute value against the currently available APIs in the device 1. It is assumed that application management software 4.9 keeps track of the available APIs the same way it keeps track of the available MIDlets. If such API is already present, application management software 4.9 will move to application provisioning phase. If the API is not present,

application management software 4.9 needs to do API provisioning before it can continue to application provisioning. The location of the required API library 3 can be read from a JavaAPIImplementation attribute, which points to an appropriate jar file (~~Java~~JAVA™ archive) in the file system of the cover 8.

--

In the paragraph beginning at p. 14, line 22:

-- If the device 1 does not have an existing ~~Java~~JAVA™ Location API, application management software 4.9 needs to configure the classloader of the VM to start using the location API library from the accessory memory 2.2. JavaAPIImplementation attribute value is appended to VM "classpath" (depending on implementation this may be something else). The application management software 4.9 is notified about the new API and "LocationAPI" entry is added to the list of available APIs of the application management software 4.9.

--

In the paragraph beginning at p. 15, line 1:

--If the device 1 already has a ~~Java~~JAVA™ Location API present by default, replacing the device Location API implementation with the Location API implementation of the cover 8 may not be possible due to security restrictions. In addition, replacing the device 1 Location API implementation may not be feasible, if the present implementation is using device features that would not be available with the Location API implementation of the cover 8.--

In the paragraph beginning at p. 17, line 5:

--Since the locator cover 8 includes a ~~Java~~JAVA™ Location API implementation, it is possible to download and execute any other third-party MIDlets that are based on this API.--

In the paragraph beginning at p. 17, line 11:

--The location application 4.13 uses ~~Java~~JAVA™ Location API for obtaining info from the cover 8.--

In the paragraph beginning at p. 17, line 14:

-- The ~~Java~~JAVA™ Location API internal implementation uses a suitable communication protocol (such as some I2C-based protocol) for communicating with the cover 8. This requires that the ~~Java~~JAVA™ environment of the device 1 is able to offer such ~~Java~~JAVA™ protocol API for the downloaded ~~Java~~JAVA™ Location API implementation.--

In the paragraph beginning at p. 17, line 22:

-- The accessory server 4.10 also notices the removal, for example, because of an interrupt generated by the hardware of the device 1. On removal, the application management software 4.9 is notified that the ~~Java~~JAVA™ Location API implementation is no longer available and installed location MIDlets cannot execute (ClassNotFoundException or VM internal error is thrown). The application management software 4.9 removes the "LocationAPI" entry from its list of available APIs.--